Development of Wind Tunnel for Laboratory Wind Turbine Testing

A Senior Project
presented to
the Faculty of the Electrical Engineering Department
California Polytechnic State University, San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science

By:

Danny Alexander Zepeda

June, 2011

Table of Contents

List of Tables and Figures

Acknowledgements:

I would like to thank everyone who has helped make this senior project possible. Firstly, I would like to thank my adviser Dr. Dale Dolan for his help and support for this project. I also would like to thank Jaime Carmo for the major contribution to this project and the construction of this project. I would like to thank Dr. John Oliver with his assistance in C programming and allowing me to use his STK600 to interface with all the components. I would also like to thank Dr. Taufik and Jeff Gerfen for all the equipment and components that they donated for this project. I give a special thanks to the College of Architectural Engineering Design where Royce Chow did a great job with the welding of the frame to house the large fan. Thank you also to Knect's in San Luis Obispo for manufacturing and donating a round to square duct to the Electrical Engineering department to be used for this project.

Abstract:

A wind tunnel with a 0.5 HP motor was purchased by Cal Poly from Hampden for $16,000. It can only simulate winds of up to 10 MPH. The tunnel has a manual dial that controls the wind speed. It also measures the voltage and current output with an analog measure display. This wind tunnel was expensive and unfortunately cannot achieve very high wind speeds. For this reason, a similar wind tunnel with more potential and with a smaller budge was proposed to be built for use in the Sustainable Energy Lab. The new wind tunnel that was built is able to simulate winds of up to 45 MPH and has an LCD display for the measurements of voltage and current. This whole project was done with a budget of less than $6,000 and used mostly salvaged and recycled materials.

Introduction:

Recently Cal Poly Electrical Power Program purchased a Wind turbine trainer for $16,000. This Hampden H- WPG-1B Power Generator is shown in Figure 1. After a couple of test runs and experimenting with the new lab equipment it was soon observed that the Wind Trainer had a 0.5 HP motor that would only generate max wind speeds of 10mph. This wind speed is not enough to achieve maximum power for most wind generators.

Figure 1. Wind Turbine Trainer Hampden H- WPG-1B

In addition, wind speeds could only be changed with a dial, and there was no way to measure wind speed. Voltage and current were measured with an analog meter. Instead of purchasing a more powerful wind trainer which would be very costly, this senior project was proposed in order to build a more powerful wind trainer. The target budget for this senior project was $6000. The cost savings created by building this senior project versus purchasing a more powerful wind turbine trainer can potentially be used to purchase other equipment and encourage the EE department to open a new Wind Power Energy Class.

After consultation with the Power Instructors and Dr. Dale Dolan, it was decided that the EE department would purchase a tube axial fan to blow wind to a wind turbine. After researching fans that produce high wind speeds, the Power Program purchased a 3.5' diameter fan that blows wind of 25,320 CFM for this project.

The 1st phase of this project was to control the fan in order to control wind speeds, change wind speeds, and digitally measure the speed of the wind. The wind trainer that was purchased from Hampton cost the EE department $16,000. The goal of this project was to keep the costs under $6,000. The major purchases for this project were the tube axial fan and any materials needed to establish the fan frame. In order to save the department money, equipment was salvaged from other projects, or from donations that had not been used. Instructors from previous courses helped with extra components to use instead of purchasing equipment. For example, we obtained the 5 hp motor, microcontrollers, keypads, LCD displays, wires and electric components in this manner.

Background

The Hampden H- WPG-1B Power Generator would only generate maximum wind speeds of 10mph which is not enough to achieve maximum power for most wind generators. Because of the limitations of the purchased wind trainer, this senior project was devised so that a wind trainer could be constructed to be used for a wind energy class. It is also a hope that this project would save the EE department money in order that they could then purchase other necessary equipment and therefore encourage the EE department to open a new Wind Power Energy Class.

Requirements

Phase 1 of this project had the following requirements:

- Choose a tube axial fan size for a 5hp motor that had been donated to the EE dept.
- Generate wind speeds from 0 – 40mph.
- Design a frame for the tube axial fan.
- Select a variable frequency drive size for the 5HP induction motor.
- Use a microcontroller to interface components.
- Measure wind speeds using an anemometer.
- Interface with a LCD display to read values and commands.
- Use a 16 key, key pad to interface with the microcontroller and LCD.
- Have the following options for project:
    - Measure wind speed
    - Change wind speed
    - Increment and decrement the motor frequency
    - Input desired wind speed

Design Processes:

Wind Generating Design:

Purchasing the correct fan was critical for this project's success. Important criteria in fan selection included shipping time, volume flow rate of the fan and space limitation. The different fans that were researched included centrifugal and axial fans. A centrifugal fan has a wheel with many blades or ribs and makes air enter from the side of the wheel. It will turn 90 degrees and it accelerates with the centrifugal forces as air flows over the fan blades to the fan housing. The current wind trainer purchased from Hampden currently uses a centrifugal fan. Figure 2 shows a centrifugal fan.

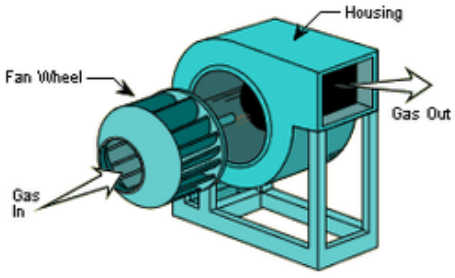


Figure 2. Centrifugal Fan

Figure 3. Tube Axial Fan

For this project a tube axial fan was used in order to achieve higher CFM. The axial fan uses blades that force air to enter in parallel to the shaft were the blades rotate. This fan is also belt driven as opposed to being direct drive. Direct drive would cost more because the motor would also have to be purchased. Since a motor that has been donated is being utilized, a belt driven fan was chosen over a direct drive. Figure 3 shows a Tube Axial Fan similar to the one that was purchased for this project.

Large Cubic Feet per Minute (CFM) was taken into consideration during research for the tube axial fan. CFM is a measure of the volume of air passing through the fan per minute. It was discovered that a tube axial fans provides the largest CFM compared to other fans. The fan used for this project was purchased from Grainger. Grainger was the only store that had tube axial fans in stock and was able to send the fan in a timely manner (two business days). Other fan manufactures would take to 6 - 8 weeks to manufacture and deliver to San Luis Obispo.

Size was also a factor in fan selection, because the larger the fan the more CFM it produces. It was important to make measurements of the department door and the Sustainable Energy Laboratory door in order to make sure that the fan that was purchased would fit into the room. The maximum width of the door involved was 34 inches.

In order to save money, an induction motor available in the EE department was used. This was the largest motor in the EE department. With all these size specifications, a tube axial fan with a 42” diameter blade span and 32”depth of fan casing was selected. Installing a 5hp motor allowed this fan to provide 25,320CFM. Finding a fan with a large CFM was important in order to produce large wind speeds. An excel spreadsheet was used in order to calculate the estimated maximum wind speed. A 33” x 33” square area for the desired camber that the wind

generator would be placed in was selected. With a 32” diameter fan that produces 25,320 CFM, it was determined that it could generate wind speeds from 35mph to 40 mph.

Computer Aided Drawing Sketch

Components that needed to be designed for this project included a frame to hold the tube axial fan, the square chamber for the wind generator and the round to square duct that connects the fan and the square chamber. Google Sketch was used to design the frame, the house of the tube axial fan and the casing to the wind turbine. Using computer aided drawing was very helpful in order to see what materials were needed to be purchased and what measurements needed to be made. Figures 4 & 5 below are rough sketches of the desired wind simulator design.

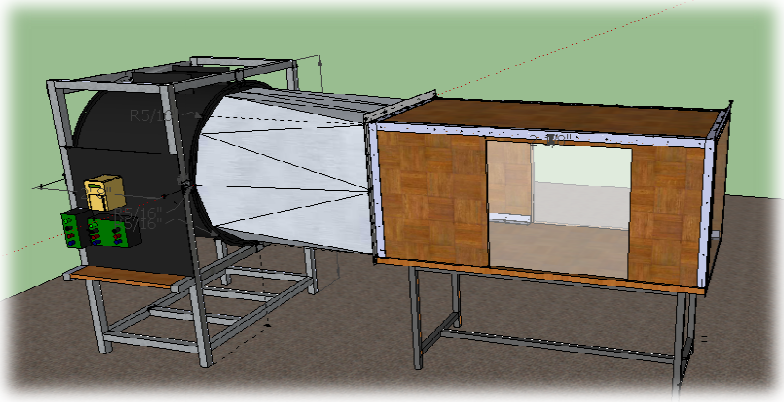


Figure 4: Wind Simulator CAD Drawing

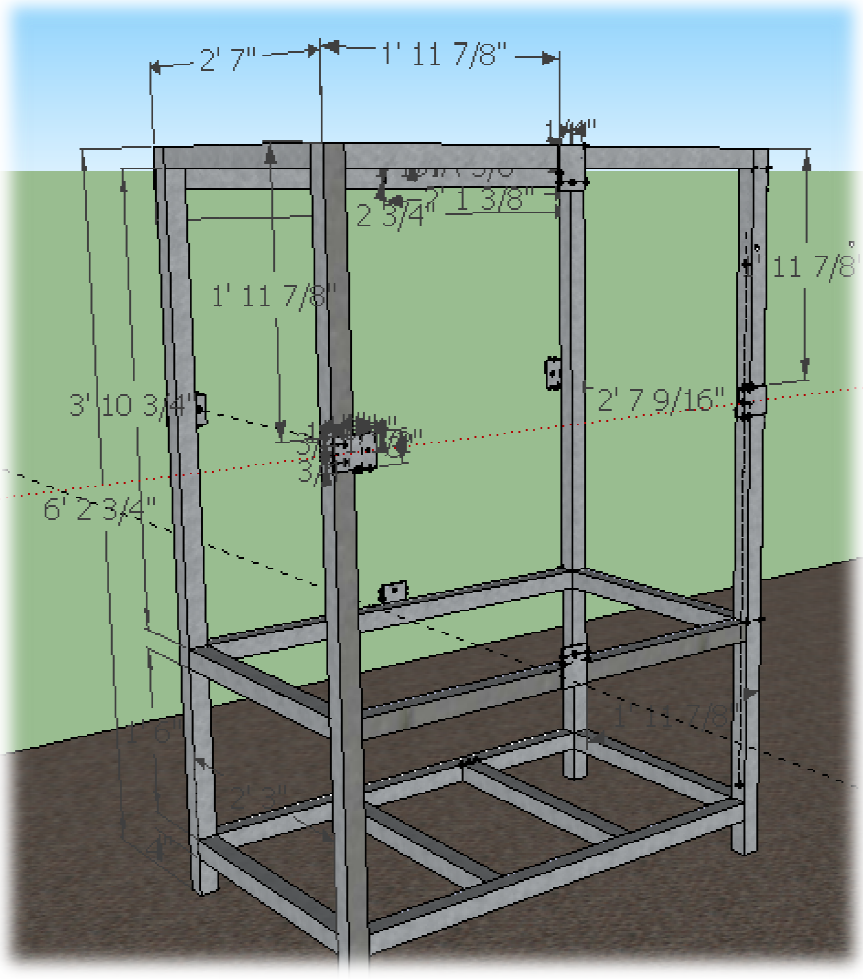


Figure 5. Wind Simulator CAD Drawing

Fan Frame Design

After designing a soft copy of the frame, the design shop at the College of Architectural Engineering helped weld the frame for the tube axial fan. Over 70ft of 2"x 2" -1/4" thick wall of tube steel was purchased from B&B Steel & Supply Co in Santa Maria for $357.60. The dimensions are shown in Figure 5.

Test Plans:

The test plans are divided into three sections:

1. Testing of the variable frequency drive control.
2. Testing of the LCD display and key pad.
3. Testing of the timer to measure the wind speed.

VFD Control Testing: After writing the code, the VFD external ports were connected to the Micro Controller. The buttons on the controller are used to start, increase, and decrease the frequency and also use LED to show which button is applied.

LCD Display and Key Pad: RS232 was tested with the computer and then after writing the program using USART register, a button on the micro controller was used to display “Hello” on the LCD display. Code was written to allow a selection of a key which displays on the LCD the character pressed. This shows that the Micro Controller is properly communicating the LCD display and the USART register.

Wind Measurement Test: After writing the code to measure the wind speed, a function generator was connected to the micro controller to provide a square pulse and apply the same square pulse to the weather station controller. The LCD display reads the same wind speed as the weather controller. This shows that the timer is correctly measuring the period of the square pulse provided by the function generator.

Development and Construction :

Figure 6 : Frame being welded together

When the tubing was cut into the correct size, the frame was tacked and stick welded together. Figure 6 shows the 2”x 2” steel tubing being welded together. When the welding was completed, the grease from steel tubing was rinsed and cleaned off. The frame was then primed and spray painted with black paint. When the paint dried, a crane lift was used to lift the tube axial fan and bolt it to the steel frame. Figure 7 shows the fan being lifted and bolted to the frame.

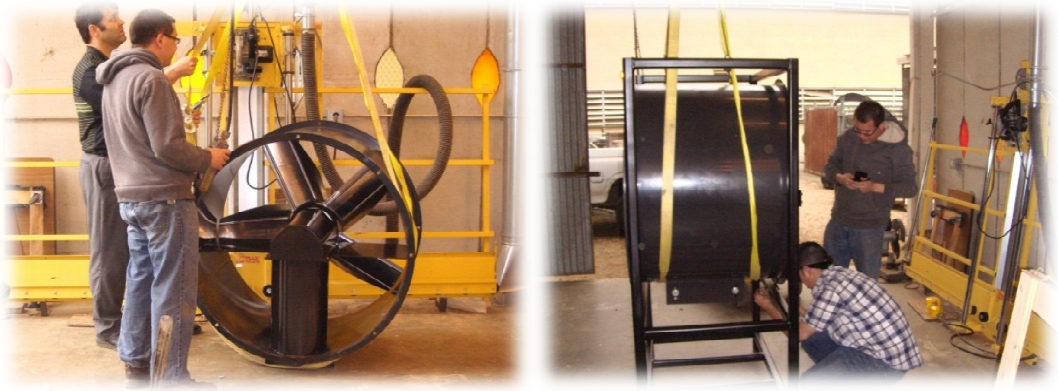


Figure 7: Bolting the fan to the steel frame

For the safety of the students, two fan guards were purchased from Grainger. The two fan guards were bolted with the fan and the steel frame. This allows protection from the fast rotating blades. Before the fan was lifted into the frame, it was decided to position the fan where the induction motor could be installed on the bottom of the fan. This lowered the center of gravity. The metal base used to position the 5Hp induction motor was removed to make it easier to bolt the motor to the frame. A car jack was then used to lift the motor with the base to position it on the fan. The protective cover of the motor and the belt was installed and tightened. Figure 8 shows the installed motor. After installing the 5hp motor, the fan was then fork lifted to building 20 and a pallet jack was used to move the project to the sustainable energy laboratory in room 150. This is shown in figure 9. Figure 10 shows the CAD drawing and the actual frame with the fan attached to it.

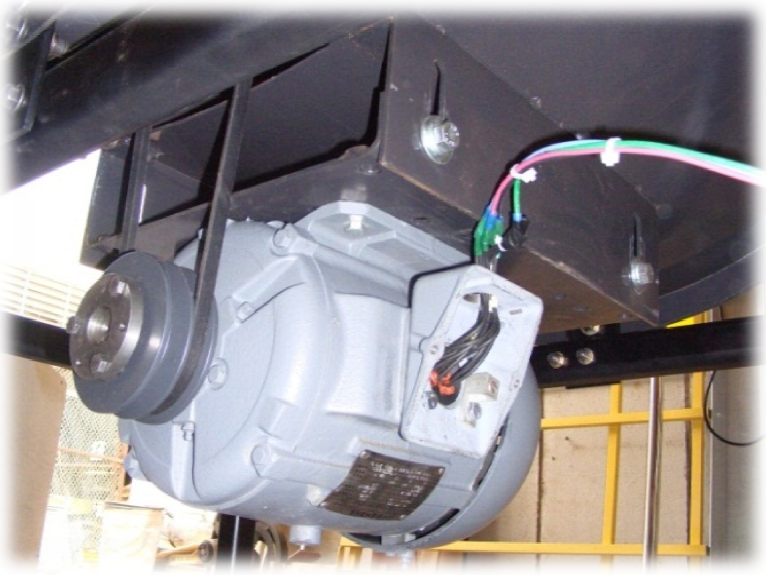


Figure 8: 5Hp Induction motor installed to the fan

Figure 9: Fan Moved Via Forklift to Building 20

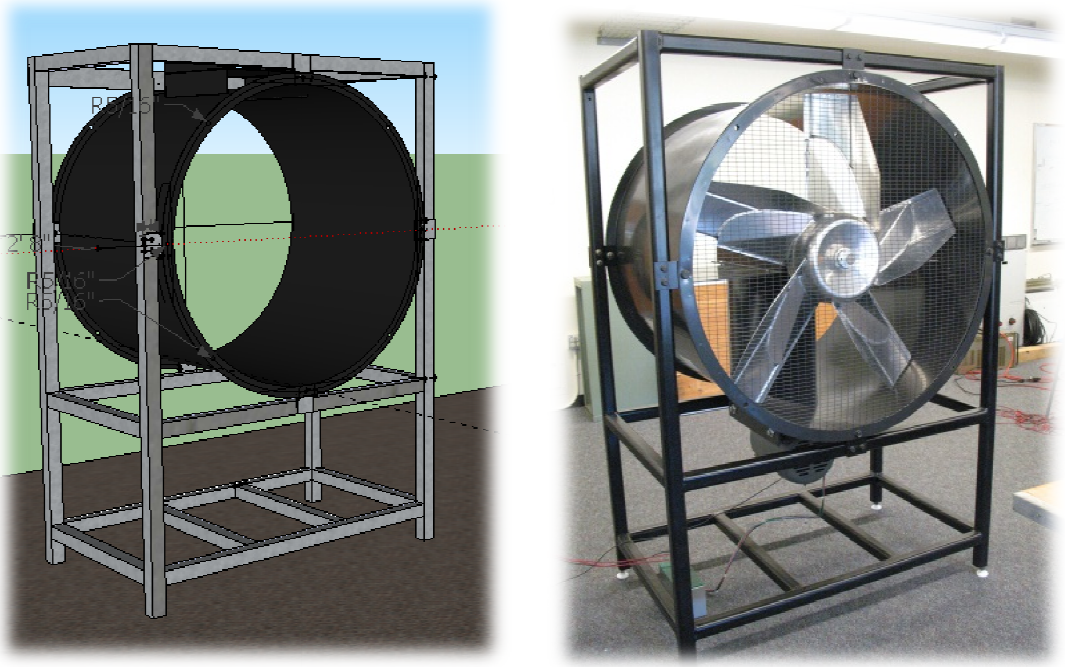
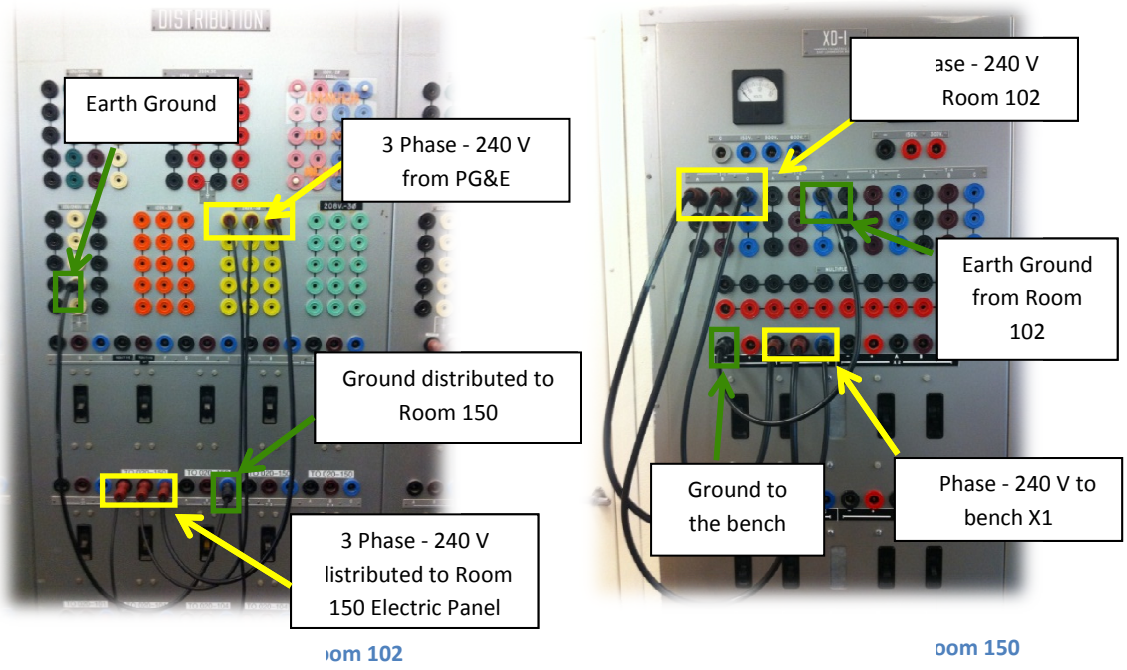


Figure 10: (Left) CAD Drawing of Fan and Frame (Right) Actual Fan and Frame completed

Motor Connection

Since the fan was positioned firmly and safely to the frame, it was now ready to hook up the VFD and complete some tests. The primary power source was supplied by PG&E in Building 20 room 102. 240 V – 3 phase power was taken from room 102 electric panels and distributed to room 150, which is shown in figure 11 below.



When 240 V, 3phase power was connected to room 150, the power was distributed to one of the lab benches in the Sustainable Energy Laboratory as is shown in figure 12. Connection boxes had to be made to protect the VFD and allow safe motor connection and interfacing. Figure 13 below shows the connection box that supplied 240 - 3 phase power from the bench and protected the VFD with 20 A fuses. The outputs of the VFD are also connected to banana lead connections. Another box connection was used to connect to the motor and to banana leads. Figure 13 also shows the connection with the VFD and the motor.

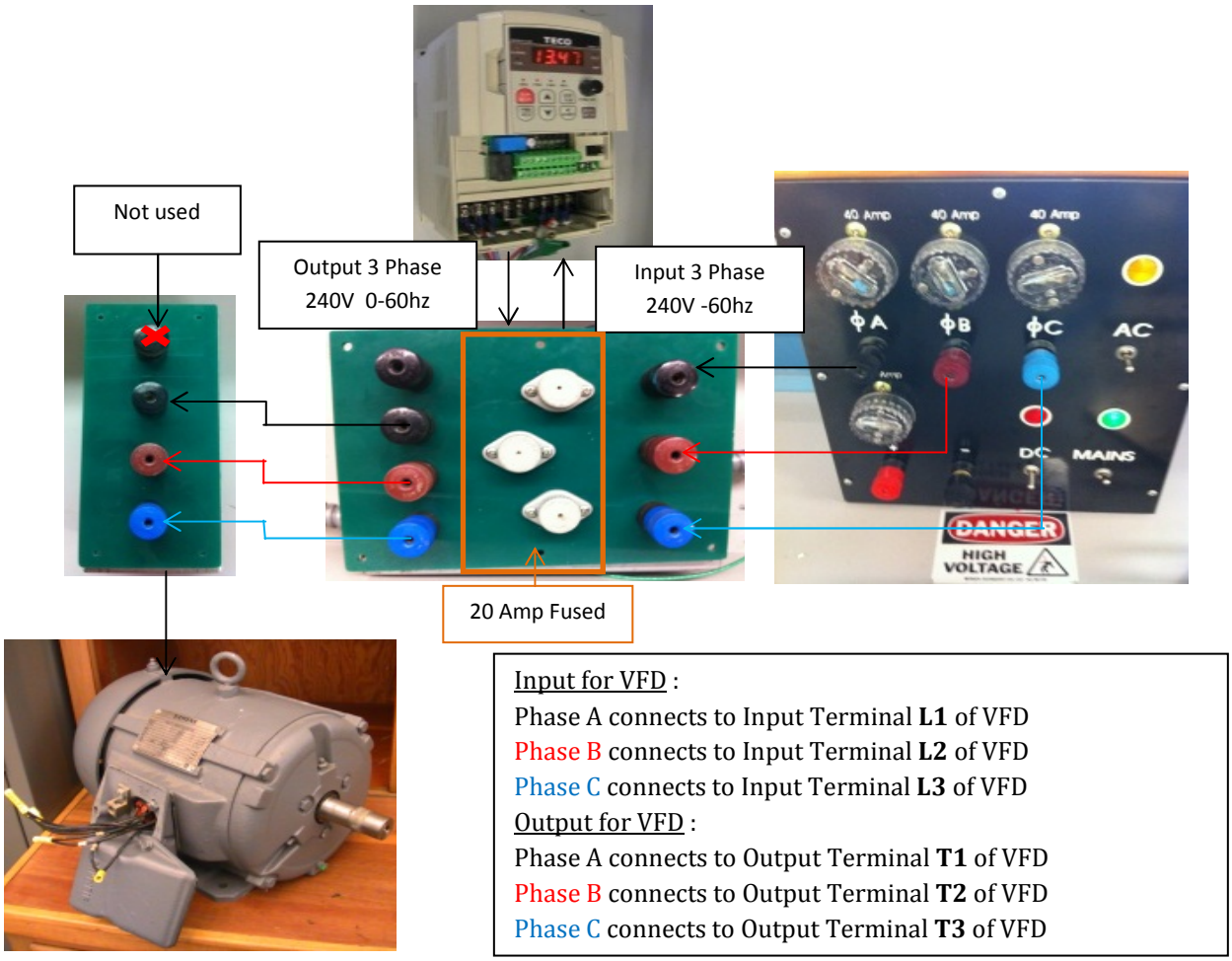
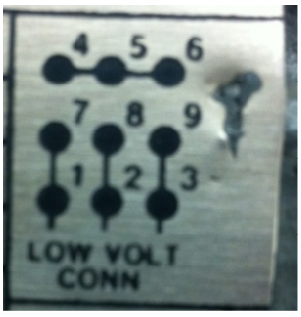


Figure 13 : VFD and Motor Connection

The motor can be connected in two different ways with low power of 240V or high power of 480V. The name plate shows the two different types of hook up. The configuration depicted in figure 14 was used since the lover voltage of 240 V was used.



Phase A connected to T1 &T7
Phase B connected to T2 &T
Phase C connected to T3 &T9

Figure 14 : Low Voltage Connection

The frame of the motor was also grounded where it can be taken from earth ground in room 102 connected to earth ground to room 150 and connected to the lab bench. The top black banana input was left unconnected for any other connections and can be used to connect to ground. Wire T4 T5 T6 was left floating. After all the wires were properly connected and proper power had been supplied to the bench, the VFD was ready to be turned on and testing of the fan and VFD was able to begin.

Integration and Test Result

Variable Frequency Drive

The Variable Frequency Drive was defaulted to the keypad and maximum frequency was 60 Hz. There are many functions that can be changed with the function button for this project. The next steps will show how to change from keypad commands to external ports commands.

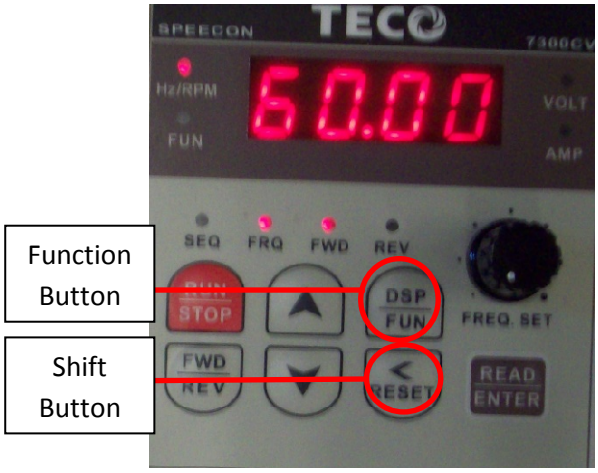


Figure 15: Key Pad of the VFD

1. Press Function Buttor 
2. Press Shift Button 3 times  until the left most digit is flashing.

3. Press the up  button to change the function code 0 (Drive Operation Mode) to 1 (Start/Stop and Frequency Control Modes).

4. Press enter  and up  to change from Key Pad(0000)to External Run/Stop Control(0001) and press Read/Enter 

This allows use of the external ports to run and stop the VFD shown in figure16.

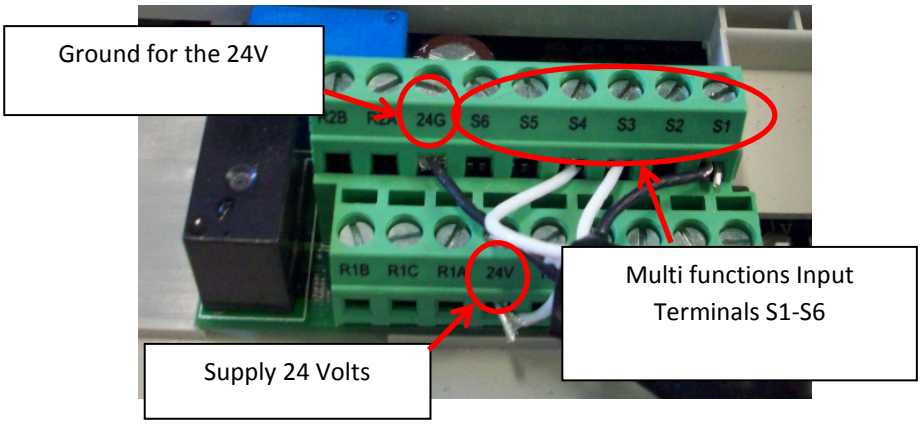


Figure 16: External terminals for MFIT

5. Press up 6 times  to change from Run Command Source Selection(1-00) to Frequency Command Source Selection (1-06).

6. Press enter  and up  to change from KeyPad(0000) to Up /Down Frequency Controlling Using MFIT (0003) and press Read/Enter )

This allows for changes to the frequency using the Multifunction's Input Terminals (MFIT) on the external ports shown in figure 16.

The Terminal can be changed in the function code 5.

Terminal S1 (5-00) set **Forward/ Stop Command (0000)**

Terminal S2 (5-01) set to **Reverse/Stop Command (0001)**

Terminal S3 (5-02) set to **Up Command (0014)**

Terminal S4 (5-03) set to **Down Command (0015)**

Now the variable frequency drive is ready to be controlled with the multifunction input terminals (external terminals shown in figure 16). Tie a wire to the 24V terminal on the external ports. To turn on the VFD, apply the 24 volts to terminal S1. To increase the VFD frequency, apply 24 V to S1 and S3 at the same time. In order to decrease the VFD frequency, apply 24V to S1 and S4 at the same time. The micro controller will be used to interface with these external ports to operate the VFD.

The STK 600 is used to control the commands of the VFD and measure and speed of the simulated wind. C programming is written to interface with the following:

- Variable Frequency Drive
- Anemometer
- Key Pad and LCD monitor

To interface with the Variable Frequency Drive regular ports are set to output on the STK600 Board. In order to interface with the anemometer a trigger timer on the STK 600 was used. A trigger timer times the period of any applied signal. The trigger timer was used to measure the period of the square pulse from the anemometer. To interface with the key pad and the LCD monitor, the STK600 USART was used to communicate with RS-232 connection of the keypad. The STK600 is equipped with an analog to digital converter which can be used to implement future sensor, this allows students to implement voltage and current measurements to the system.

The variable frequency drive can be controlled by various options.

1. With the keypad on the screen.
2. With the potiometer.
3. Via external connections.
4. With the analog connection.

By default, the variable frequency drive is controlled by the keypad. Referring to the user manual helps to obtain a good understanding of how to control the different functions of the VFD.

Terminal S1 is set up for Start/ Stop,

Terminal S3 in set up to Increment Frequency,

Terminal S4 is set up to Decrement Frequency,

Set up for these functions where described in the previous section.

The goal is to apply 24V to pin S1 to start the VFD and when you de-apply the pin the VFD should stop.

When you apply a constant voltage to S1 and begin tapping S3 at the same time it should increment the frequency. And with 24V apply to S1 and you begin tapping the S4 the frequency should decrement.

The problem that was encountered was that the micro controller didn't apply 24 volts, so a switch circuit was created shown in figure25. Opto Isolators (CNY17F-1) were used to isolate from the VFD. This circuit should act as a 2V to 24V switch shown in figure 17.

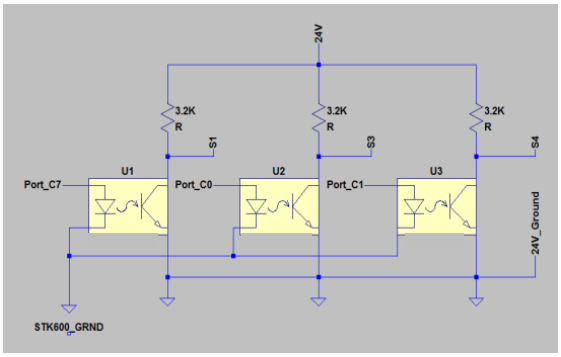


Figure 17: 2V to 24V Switch Circuit

The program was created to have the micro controller apply voltage to turn on the VFD and to increment and decrement the frequency.

The simple code can be written to turn on and off with the STK micro controller:

*Pressing **btn7** will turn on the VFD where Port C7 will apply voltage to S1.*

*Pressing **btn6** turn the VFD off*

*Pressing **btn1** increased frequency by connecting port H0 to Terminal S3 of VFD.*

*Pressing **btn0** decrease the frequency by connecting port H1 to Terminal S4 of VFD*

```
//output/input and delay header files
#include<avr/io.h>
#include<util/delay.h>

int main(void)
{       //Variable Declaration
        int switch1_flag = 0; //- Used to enable/disable
        int switch2_flag = 0; //- Used to enable/disable
        int count = 0; //- Used to enable/disable
```

```
DDRA = 0x00; //Data direction register port A set to input Buttons!!!
DDRB = 0xff; //Data direction register port B set to output LEDS
DDRC = 0xff; //Data direction register port C set to output to VFD

//Turns all LEDs ON LOW = ON

PORTB = ~0x40;// LED 6
PORTC = ~0x40;

while(1)
{
        if (PINA == 0x7F) // Start (Btn 7) output 24 V to S1
        {       PORTB = ~0x80;// LED 7
                PORTC = ~0x80; // apply  24V to S1
                switch1_flag = 1; // allows to be in the while loop and
                                  //keep S1 with high 24V

        ////////////////////////////////In Run Mode//////////////////////////

                while(switch1_flag ==1)
                {
                        if (PINA == 0xBF) // Stop (Btn 6) output 0V to S1
                        {       PORTB = ~0x40;// LED 6
                                PORTC = ~0x40;
                                switch1_flag =0; // Takes out of run mode
                        }


else if(PINA == 0xFD)// Up Frequency Button 24V to S3
        {

                count=0;


                while(count != 1)       // Changing the value 1 change the
                        //frequency increments
               {

                        PORTB = ~0x82;  // LED7 and LED 1
                        PORTC = ~0x82;
                        _delay_ms(500);
                        PORTB = ~0x80;   // LED7
                        PORTC = ~0x80;
                        _delay_ms(1000);
                        count++;
```

```
                }
        }


else if(PINA == 0xFE) // dwn Frequency Button 24V to S4
        {       count=0;
                while(count != 1)       // Changing the value 1 change the
                                        //frequency increments
                {
                        PORTB = ~0x81;  // LED7 and LED 0
                        PORTC = ~0x81;
                        _delay_ms(500);

                        PORTB = ~0x80; // LED7
                        PORTC = ~0x80;
                        _delay_ms(1000);

                        count++;

                        }

                }
         }

}
return 0;
} //END
```

VFD Test Results: After selecting btn 7,VFD was able to be turned on. Then the frequency was incremented and decremented by selecting btn0 and btn1. The results were satisfactory and code could then be added. The next step was to interface with LCD and the keypad.

Interfacing with the Key Pad/ LCD

The next step was to control the LCD and the key pad. An RS232 connection was used which was first checked with the computer in order to determine if the key pad was working. The following are the steps involved:

1. Power up the display with 5V power source, Pin out is shown in figure 18.

Figure 18: Power source pin out for LCD Display

2. Connect the display to the PC with a RS-232 cable.
3. Open Hyperlink on the computer.
4. Select the Baud Rate of 19,200.
5. Press a key on the keypad. An upper case ASCII Character should display on the screen.

   If a key on the computer's keyboard is pressed, it should display on LCD display.

The keypad layout is shown in the figure 19 down below:

	Columns				
R		1	2	3	4
O	1	**A**	**B**	**C**	**D**
W	2	**F**	**G**	**H**	**I**
	3	**K**	**L**	**M**	**N**
S	4	**P**	**Q**	**R**	**S**

Figure 19: Key Pad Lay Out

The next step was to be able to connect the RS 232 with the microcontroller. A USART was used which allows to communicate bi-directional by setting up the transmission and receiving functions in the C code. Set up for transmission and receiving functions is provided, and can be broken up into three steps.

```
//Setting up the UASRT
#define FOSC 8000000 // Clock Speed at 8MHz
#define BAUD 19200   // From the Keypad/LCD Data sheet
 #define MYUBRR ((FOSC/16/BAUD)-1)// UBRRN is 51
#include <avr/io.h>
#include<util/delay.h>

void USART_Init( unsigned int ubrr)
{          UBRR0H = 0;     // (unsigned char)(ubrr>>8);  Setting Baud Rate High Registers pg 227

        UBRR0L =(unsigned char)ubrr;   // 0x67;
                // Setting Baud Rate Low Registers pg 227   // USART Control and Status Register
        nB Pg 224
                UCSR0B =(1<<RXEN0)|(1<<TXEN0); // Bit 4 Receiver Enable and Bit 3 Transmitter
        Enable
                UCSR0C =(3<<UCSZ00);//(1<<UCSZ01)|(3<<UCSZ00); // Set Frame format: 8
        data, 2 stop bits
                // (0<<USBS0) USBS is Stop bit select ( 0=1bit, 1=2bit) // UCSZ Character Size  3 =
        011 = 8bit
        }
        ////////////////////////////////////////////////////////
                Void USART_Transmit(unsigned char data)
                {       while(!(UCSR0A & (1<<UDRE0))) //Wait for empty transmit buffer
                        {          ;       }
                //Put data into buffer, sends the data
                UDR0 = data; // data sent out
                }
        ////////////////////////////////////////////////////////
        Unsigned char USART_Receive(void)
                {   while( !(UCSR0A & (1<< RXC0)))
                        {         ;          }
                        return UDR0;
                }
        //Example of Code being outputted Use USART_Receive() when you would like to
        receive an input from
```

```
//the keypad and use USART_Transmit() when output to the LCD display :
        int main (void)
        {         char data; //Use data to store value
                   USART_Init(MYUBRR); // goes back and set the frame up with my brad
value
                  while(1)
                  {         data=USART_Receive();  //Press Key and should display on LCD

                            USART_Transmit('[');
                            USART_Transmit(data);
                            USART_Transmit(']');
                            //Print Hello on LCD Display
                            USART_Transmit('H');
                            USART_Transmit('E');
                            USART_Transmit('L');
                            USART_Transmit('L');
                            USART_Transmit('O');
                  }

        }
```

Connections for the STK 600:

Register 0 was used for USART in the code. So in order to connect to the RS-232 connection, RS-232 Spare Rx would have to be connected to Port PE0 and Spare Tx to Port PE1 which is designated for the RS-232 as shown in figure 20 below.

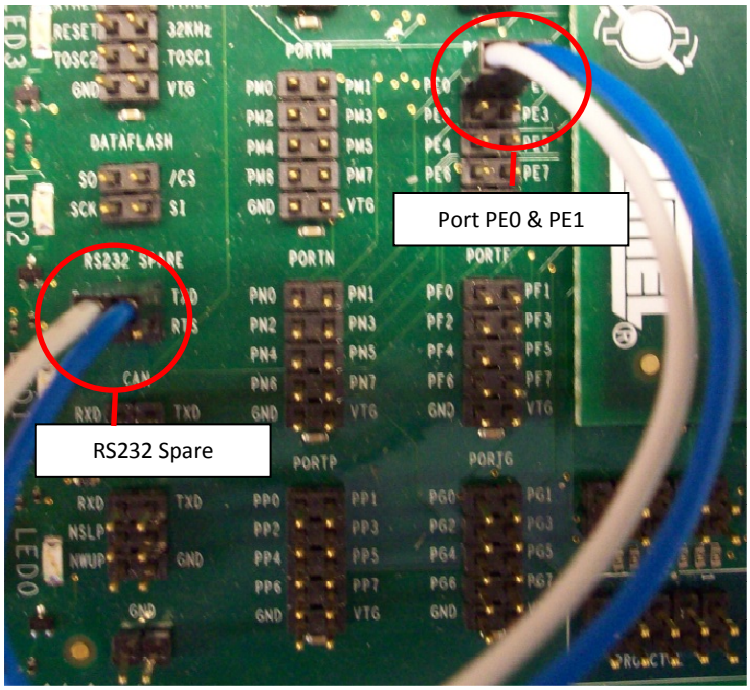


Figure 20: Spare RS232 Pin out

Notes:

As soon as the code was connected, the correct cable needed to be connected. A Null cable was used which switches the 2 pins around.

Later I discovered that an interrupt would be needed in my code. Down below the ISR code is provided to allow my code to keep polling for an input of the keypad. So when a key is pressed the code interrupts and stores the value in press_function variable. So instead of the code to wait for a key to be pressed like when the code used data=USART_Receive(); it will just

interrupt the code when a key is pressed.

```
ISR(USART0_RX_vect)
  { press_function = UDR0;
  }
```

With the code used “Hello” was able to be displayed on the LCD display. A key was also able to be pressed to output that character to the LCD display. This demonstrates communication with the micro- controller and LCD display. Code can now be written that can provide a menu to interface with the user and also display wind speed when ready.

Interfacing with Wind Sensor

Interfacing with the Wind Sensor is the most important part of the project. The anemometer takes a 12 V source to power up the weather controller and it supplies a 5 V source to one of the pins of the phone jack. As the anemometer spins, it shorts the 5V with a diode and produces a square pulse. Set up for the weather station is shown in Figure 21. The faster the anemometer spins, the shorter the periods.

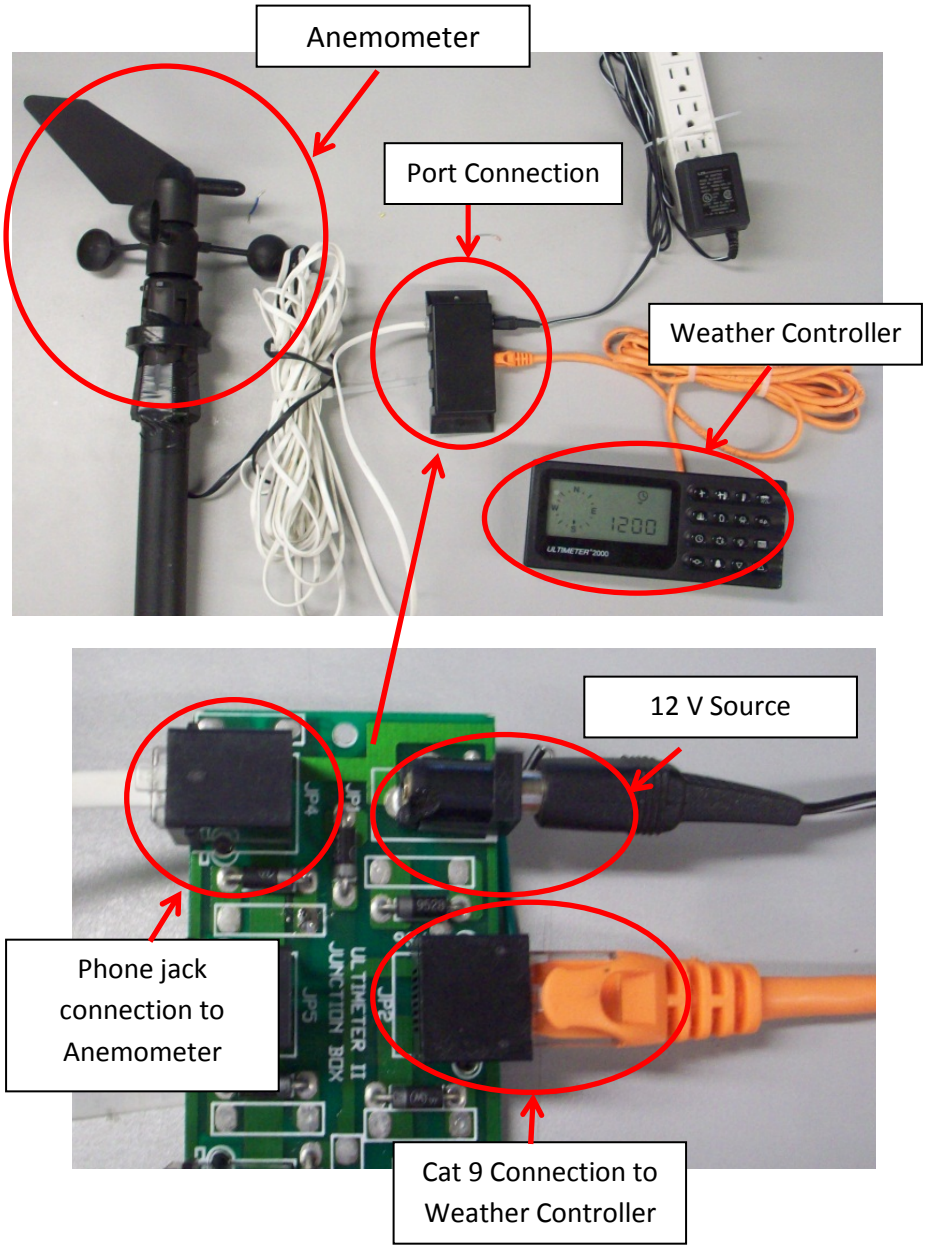


Figure 21: Weather Station Setup

The controller outputs wind speed when the anemometer button is pressed as is shown in figure 22.

Figure 22: Controller

Figure 23 shows a green wire that was solder to the pin that provides a square pulse of the anemometer. The black wire is used for the ground. These wires are connected to the micro-controller to provide interfacing with the anemometer.

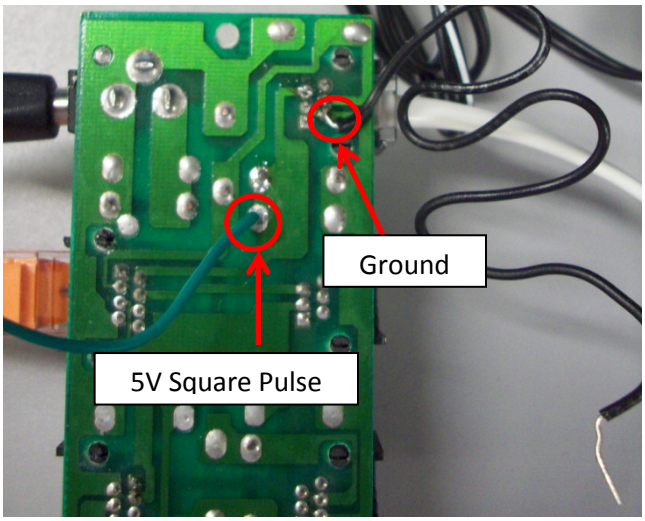


Figure 23: Solder Wires for Square Wave Output

A timer on the micro-controller was selected to be used to measure the period of the square pulse. Time Register 1 was used which corresponded to port PD4. The square impulse would be inputted at port PD4. The following code was used to set up the timer capture which starts the timer as soon as it sees an interrupt of a rising edge.

```
//////////////////START OF CODE//////////////////////
#define F_CPU 8000000//CLK speed
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
volatile unsigned int gvintT1Count = 0;
unsigned long int Frequency_top = 0xFFFF;

        void init_clk_timers()//Timer set for max wind speed of 40mph
        {       TCNT1 = 0;
                TCCR1A=0x00;
                //Mode Selection and Prescaler settings of 8Mhz/2(65535) = 61.0361 =
        16.3838ms
                TCCR1B=(0<<CS12)|(0<<CS11)|(1<<CS10);
                TIFR1=(1<<TOV1)|(1<<ICF1); // TOV1 is timer overflow flag  ICF is timer  input
        capture flag
                TIMSK1= (1 <<TOIE1)|(1 <<ICIE1); // TOIE is timer  overflow interrupt enable
                                                 //ICIE is timer  input capture interrupt enable
                DDRD &=~(1<<PD4); // port that signal will be inputted
        }

                ////////////////MAIN/////////////////////////////////
                        int main(void)
                        {       init_clk_timers();
                                sei();
                                int count = 0; //Used to enable/disable
                                while(1)
                                {  total_2 =avg_pw*2; // Records the Speed from the ISR
                        ////////////////// Increments of 1 mph ////////////////////

                        //5MPH
                        if( (total_2<0x4A7680)  && (total_2>0x3548A0))
                        {       PORTA = ~0x01;
                                _delay_ms(500);
                        }
                        //6MPH
```

```
else if( (total_2<0x3548A0)&& (total_2>0x295220))
{       PORTA = ~0x02;
        _delay_ms(500);
}
//7MPH
else if( (total_2<0x295220)  && (total_2>0x21C0A0))
{       PORTA = ~0x04;
        _delay_ms(500);
}
//8MPH
else if( (total_2<0x21C0A0)  && (total_2>0x1C9080))
{       PORTA = ~0x08;
        _delay_ms(500);
}
//9MPH
else if( (total_2<0x1C9080)  && (total_2>0x18B820))
{       PORTA = ~0x10;
        _delay_ms(500);
}
//10MPH
else if( (total_2<0x18B820)  && (total_2>0x15BA80))
 {      PORTA = ~0x20;
        _delay_ms(500);  }
} //End While
} //End Main
//ISR for timer overflow that will be used for calculating //the speed.
        ISR(TIMER1_OVF_vect)
        {   gvintT1Count++; }
// This interrupt will start the timer every time where there is a rising edge.
ISR(TIMER1_CAPT_vect)
{       total_1 = gvintT1Count * Frequency_top
        //gvintT1Count is a counter that keeps track of  how many times the
timer overflows.
        //Frequency_top is set to value of 0xFFFF
        pw = ICR1 + total_1;
        //ICR1 is the value of the timer and then added to the //value of
overflows.
        data[record] = pw;
        // This is set up to store values to get an average
        record++;
        // Clears TCNT1 Register and ready to count again
        TCNT1 = 0;
        // Clears overflow counter & total_1 to be used again
```

```
                gvintT1Count= 0;
                total_1=0;
                 // Set to store 20 values
                record = record%20;

                //Calculated the average
                pw_total=data[0] + data[1] + data[2 ]+ data[3] + data[4] + data[5] +
                data[6] +
                data[7] + data[8] + data[9] + data[10] + data[11] + data[12] + data[13] +
                data[14]
                + data[15] + data[16] + data[17] + data[18] +data[19];

                //avg_pw is the average that will be used in the main
                avg_pw = pw_total / 20;
        }
```

To determine the period corresponding to wind speed, the anemometer was disconnected and a function generator was connected to the pin that supplied the square pulse. This is depicted in figure 24. A square pulse with a 1.22s period was applied and the weather controller was read which outputted a wind speed of 5mph. The period was decreased to .873ms until the weather station changed from 5mph to 6mph. The period of the function generator was then recoded on a excel sheet where it then calculated the frequency. The excel sheet can be shown on the next page. Time period vs Wind Speed graph and the Frequency vs Wind Speed graph can be shown in figure 25 and 26.

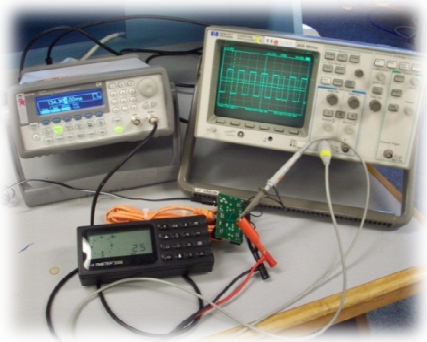


Figure 24: Simulating Wind with Function Generator

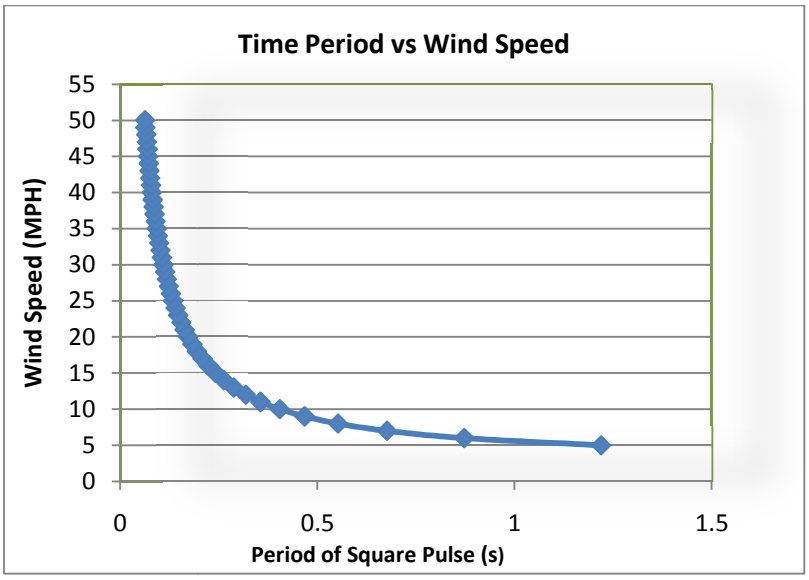


Figure 25: Time Period vs Wind Speed Graph

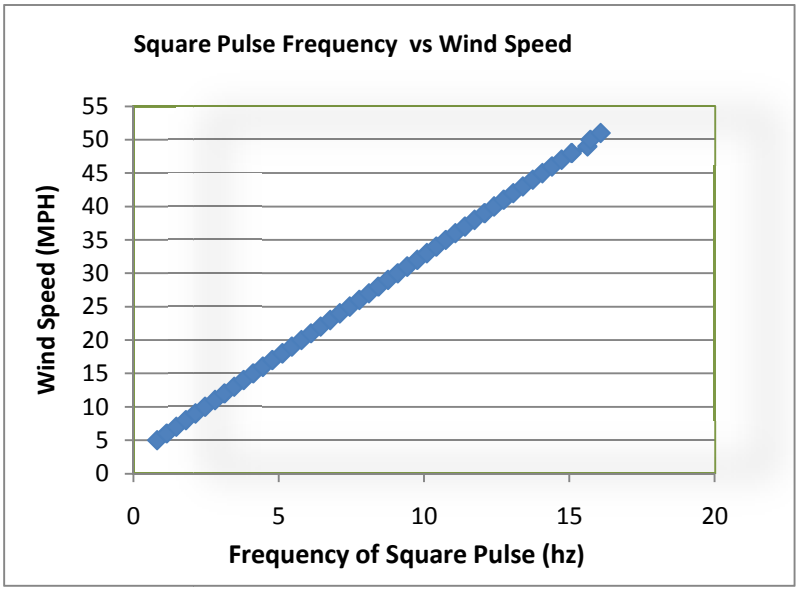


Figure 26: Frequency vs Wind Speed Graph

Table 1: Excel Table Wind Speed Values, Timing and Frequency

MPH	T1(s)	T2(s)	f1(hz)	f2(hz)	TimerValue_1	TimerValue_2	Hex 1	Hex 2
5	1.22	0.874	0.819672	1.144165	4880000	3496000	4A7680	355840
6	0.873	0.678	1.145475	1.474926	3492000	2712000	3548A0	2961C0
7	0.677	0.554	1.477105	1.805054	2708000	2216000	295220	21D040
8	0.553	0.469	1.808318	2.132196	2212000	1876000	21C0A0	1CA020
9	0.468	0.406	2.136752	2.463054	1872000	1624000	1C9080	18C7C0
10	0.405	0.357	2.469136	2.80112	1620000	1428000	18B820	15CA20
11	0.356	0.32	2.808989	3.125	1424000	1280000	15BA80	138800
12	0.319	0.289	3.134796	3.460208	1276000	1156000	137860	11A3A0
13	0.288	0.2635	3.472222	3.795066	1152000	1054000	119400	101530
14	0.2634	0.2425	3.796507	4.123711	1053600	970000	1013A0	ECD10
15	0.2424	0.2245	4.125413	4.454343	969600	898000	ECB80	DB3D0
16	0.2244	0.2091	4.456328	4.782401	897600	836400	DB240	CC330
17	0.209	0.1951	4.784689	5.125577	836000	780400	CC1A0	BE870
18	0.195	0.18334	5.128205	5.454347	780000	733360	BE6E0	B30B0
19	0.18333	0.1729	5.454645	5.78369	733320	691600	B3088	A8D90
20	0.1728	0.16356	5.787037	6.113964	691200	654240	A8C00	9FBA0
21	0.16355	0.1552	6.114338	6.443299	654200	620800	9FB78	97900
22	0.15519	0.14765	6.443714	6.772773	620760	590600	978D8	90308
23	0.14764	0.140744	6.773232	7.105099	590560	562976	902E0	89720

MPH	T1(s)	T2(s)	f1(hz)	f2(hz)	TimerValue_1	TimerValue_2	Hex 1	Hex 2
24	0.140743	0.134305	7.105149	7.445739	562972	537220	8971C	83284
25	0.134304	0.128627	7.445795	7.774418	537216	514508	83280	7D9CC
26	0.128626	0.123379	7.774478	8.105107	514504	493516	7D9C8	787CC
27	0.123378	0.118585	8.105173	8.43277	493512	474340	787C8	73CE4
28	0.118584	0.114125	8.432841	8.762322	474336	456500	73CE0	6F734
29	0.114124	0.10996	8.762399	9.094216	456496	439840	6F730	6B620
30	0.10995	0.10611	9.095043	9.424182	439800	424440	6B5F8	679F8
31	0.10612	0.10238	9.423294	9.767533	424480	409520	67A20	63FB0
32	0.10237	0.099044	9.768487	10.09652	409480	396176	63F88	60B90
33	0.099043	0.095935	10.09662	10.42372	396172	383740	60B8C	5DAFC
34	0.095934	0.093	10.42383	10.75269	383736	372000	5DAF8	5AD20
35	0.09299	0.09026	10.75384	11.0791	371960	361040	5ACF8	58250
36	0.09025	0.08764	11.08033	11.41031	361000	350560	58228	55960
37	0.08763	0.08519	11.41162	11.73847	350520	340760	55938	53318
38	0.08518	0.08274	11.73985	12.08605	340720	330960	532F0	50CD0
39	0.08273	0.08055	12.08751	12.41465	330920	322200	50CA8	4EA98
40	0.08054	0.07846	12.41619	12.74535	322160	313840	4EA70	4C9F0
41	0.07845	0.07652	12.74697	13.06848	313800	306080	4C9C8	4ABA0
42	0.07651	0.07461	13.07019	13.40303	306040	298440	4AB78	48DC8
43	0.0746	0.07278	13.40483	13.74004	298400	291120	48DA0	47130

MPH	T1(s)	T2(s)	f1(hz)	f2(hz)	TimerValue_1	TimerValue_2	Hex 1	Hex 2
44	0.07277	0.07107	13.74193	14.07063	291080	284280	47108	45678
45	0.07106	0.06944	14.07261	14.40092	284240	277760	45650	43D00
46	0.06943	0.06787	14.403	14.73405	277720	271480	43CD8	42478
47	0.06786	0.0664	14.73622	15.06024	271440	265600	42450	40D80
48	0.0663	0.065	15.08296	15.38462	265200	260000	40BF0	3F7A0
49	0.064	0.06358	15.625	15.72822	256000	254320	3E800	3E170
50	0.06359	0.0621	15.72574	16.10306	254360	248400	3E198	3CA50

Calculations to determine the Time Value that correspond to the Wind Speed:

Clk Speed		8000000
16Bit Value		65535
Pre scale		1
Timer(hz)		61.03609
Timer(s)		0.016384

One Bit time	0.00000025

$$Timer\ \ Frequency\ Value\ = \frac{Clk\ Speed}{2x\ 16Bit\ Value\ x\ PreScale}\ (\text{hz})$$

$$Timer\ Period\ Value\ = \frac{1}{Timer\ \ Frequency\ Value}\ (\text{s})$$

$$One\ Bit\ time\ = \frac{Timer\ Period\ Value}{16Bit\ Value}\ (\text{s})$$

$$Timer\ Value\ = \frac{T1(s)}{One\ Bit\ Time}$$

Convert this value to Hex and use in

Wind Sensor Test Result:

LED where used to represent the wind speed for the code shown above. A square pulse was applied to the weather controller and to the micro controller. LED 1 was set to 5mph and LED2 was set to 6mph and so on, all the way to LED 5 which corresponded to 10 mph. The LED accurately corresponds to the speed that the weather controller displayed. As well, instead of the LEDs to output wind speed the LCD display was also used to output this measurement.

Complete System Integrated:

The following was all integrated together into a program:

1. Control the Variable Frequency Drive with the Micro Controller.
2. Interface with a 16 key- key pad and the LCD display.
3. Timers to time the period of the square pulse from the anemometer and correspond it with the wind speed value.

The next step was to integrate all of the components and programming codes together. The keypad and LCD display was used to interface and turn on the fan, increase the frequency, and decrease the frequency, and also select options to measure wind speed and input a desired wind speed. Using the timer values the wind speed can be measured.

Integrating was the difficult part and involved lots of coding. The coding provided was used to integrate the control of the VFD, the 16 key keypad, LCD Display and usage of timers to measure wind speed. The code is listed on the appendix page.

System Integration Test Result:

After writing the code, the LCD display was connected to the board and the output to the anemometer. The function generator was then used to simulate wind. The LCD display output wind speed that would correspond to the wind speed value on the weather station. Then the motor was connected to the VFD and the anemometer was used to provide a square pulse. Unfortunately the wind speeds measured with the micro controller did not match the weather station wind speed values and output the wrong values to the LCD display. After some trouble shooting, it was discovered that the problem that was interfering with the timer values was Electro Magnetic Interference.

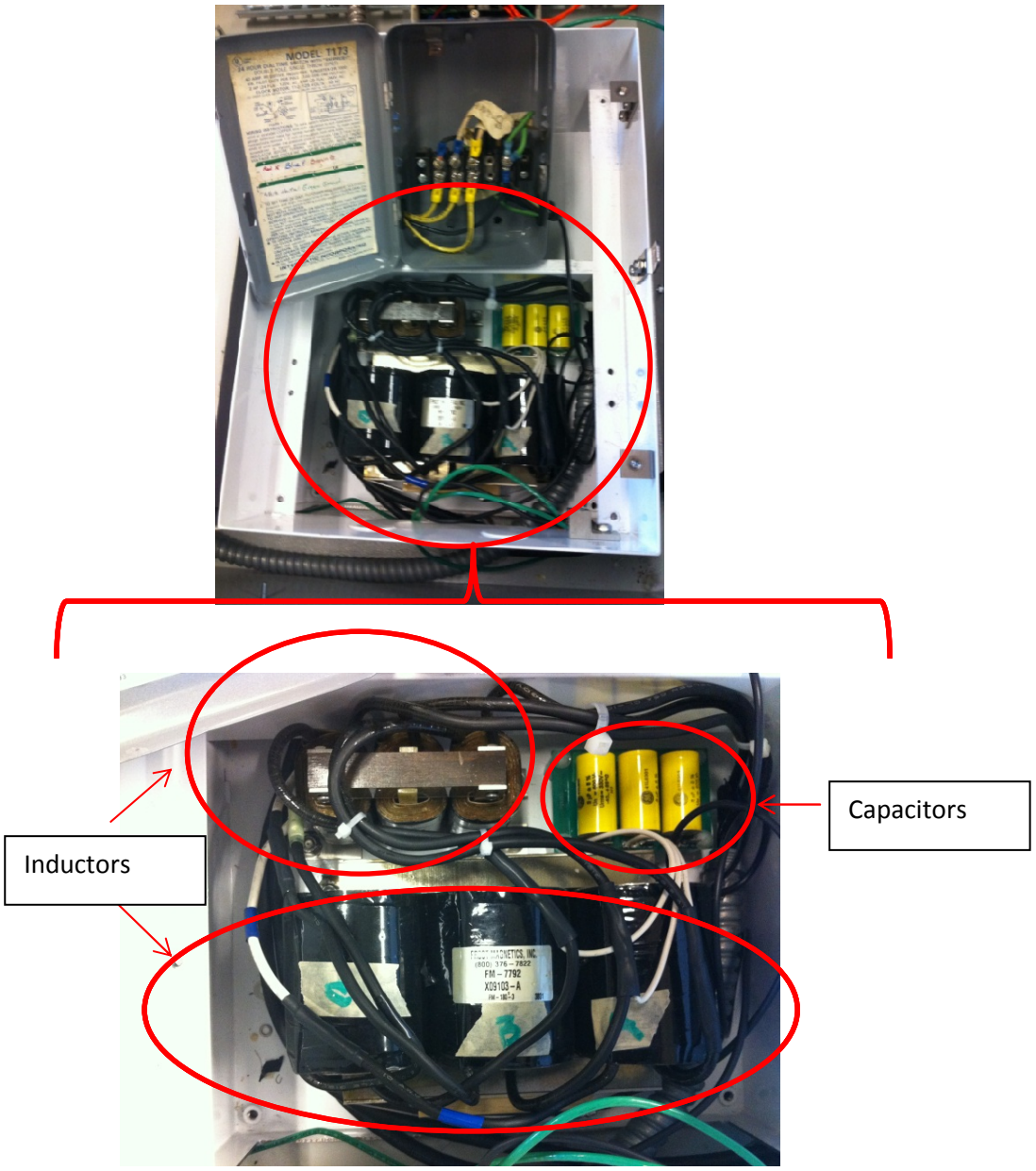
Electro Magnetic Interference (EMI)

EMI, also called radio frequency interference is a disturbance that affects electrical circuits due to electromagnetic induction or electromagnetic radiation. The Variable Frequency Drive generated radio frequency interference in the ranges of .5MHz to 1.7MHz and EMI in the ranges from 1.7MHz to 30 MHz. The Variable Frequency Drive manual provided a few solutions to minimize any EMI problem. There are a couple of things that were added to this project to minimize the interference with the micro controller:

Metal conduit shown in figure 27 was used for the 3 Phase wiring. The conduit was also grounded. Conduit was used to input the voltage to the VFD and was also used as an output to the motor.

Figure 27: Metal Conduit

An EMI Filter was recommended to be added, which is an LCL filter shown in figure 28. This is like a low pass filter. Fortunately there was a three phase LCL filter donated by a Solar Generator company available for use.



Metal housing shown in figure 29 was recommended to house the VFD. A circuit disconnect was used to house the VFD Inductors and the connection.

Figure 29: Breaker, Disconnect Box

Conclusion

Each individual component of this project was successfully built, however EMI prevented the integration of all parts of the project together. Even though EMI was minimized, it was not eliminated entirely. Possibly some more EMI shielding could improve this project, allowing all individual components to work together. Hopefully future students will continue to build on this project in order to improve the wind simulator.

Bibliography

[1] Nolan Clark, August 25,2010 , "Getting Started." San Luis Obispo, California.
(PDF Version of document downloaded February 1, 2011).

[2] November,2005, “Instruction manual TECH Inverter7300CV” Taiwan,(PDF Version of document downloaded February 21, 2011).

[3] “Solution of EMI Problems from Operation of Variable-Frequency Drive”. World Web, 28 May. 2011.<http://www.pge.com/includes/docs/pdfs/mybusiness/customerservice/energystatus/powerquality/vfd_emi.pdf>

Appendix A Part List and Cost

Donated Equipment:

STK 600 Micro Controller
5 HP Induction Motor
Wind Turbine
Key Pad and LCD Display
Metal Conduit
Wooden Table
Round to Square Duct
EMI Filter
Breaker Box
Optical Isolator
Prototype Boards
Electrical Wire

Purchased Equipment and Cost:

(1) Tube Axial Fan 42inch	$2,122.20
(1) Sheave, Two Grove	$49.61
(1) Bushing, Split Taper	$12.35
(1) V Belt 77 inch	$14.20
(2) Fan Guard	$523.36
(1) Variable Frequency Drive	$200.00
(4) 2 x 2 x .250 H.S.T. 20’	$357.60
(1) ¼ X 3” H.R. Flat 96”	$25.00
(2) 2x2X 1/8” H.R. Angle 20’	$47.19
Total :	$3,351.51

Appendix B Program Listing

```
//////////////////////////////////////START OF CODE//////////////////////////////////////

    //CLK speed
    #define F_CPU 8000000
    #include <avr/io.h>
    #include <util/delay.h>
    #include <avr/interrupt.h>
    #include <stdio.h>

    #define FOSC 8000000 // Clock Speed at 8MHz
    #define BAUD 19200   // From the Keypad/LCD Data sheet
    #define MYUBRR ((FOSC/16/BAUD)-1) // recomended UBRRN IS 51

    volatile unsigned int gvintT1Count = 0;
    unsigned long int time_stamp_1= 0;
    unsigned long int time_stamp_2 = 0;
    unsigned long int Frequency_top = 0xFFFF;
    unsigned long int total_1 = 0;
    unsigned long int total_2 = 0;

    float data[20];
    unsigned int record ;
    unsigned long int pw = 0;
    unsigned long int pw_total = 0;
    volatile long int avg_pw = 0;
    volatile char press_function =' ';
    ////////////////////////////////////////
    ////////////////////////////////////////
    int opt_A = 0; // Flag for option A
    int menu_flag = 0;
    ////////////TEXT BANK/////////////////
    int i =0;
    //introduction

    char calpoly[20] = {' ',' ',' ',' ',' ',' ',' ','C','a','l',' ','P','o','l','y',' ',' ',' ',' ',' ',' ',' ',' '};
    char lbd[20] ={' ',' ',' ','L','e','a','r','n',' ','B','y',' ','D','o','i','n','g',' ',' ',' '};
    //menu
    char Sel_Opt[20] = {'S','e','l','e','c','t',' ','O','p','t','i','o','n',' ',' ',' ',' ',' ',' ',' ',' ',' '};
    char Options_A[20] = {'A',')','M','e','a','s','u','r','e',' ','W','i','n','d',' ','S','p',' e',' e',' d'};
    char Options_B[20] = {'B',')','I','n','p','u','t',' ','W','i','n','d',' ','S','p','e','e','d',' ',' '};
    char Options_C[20] = {'C',')','T','u','r','b','i','n','e',' ','M','e','s','u','r','m','e','n','t','s'};
    ////////////////////////////////////////////////////////
```

```
char wind_speed[10] = {'W','i','n','d','S','p','e','e','d',' '};
char mph_5[7] = {'5',' ','M','P','H',' ',' '};
char mph_6[7] = {'6',' ','M','P','H',' ',' '};
char mph_7[7] = {'7',' ','M','P','H',' ',' '};
char mph_8[7] = {'8',' ','M','P','H',' ',' '};
char mph_9[7] = {'9',' ','M','P','H',' ',' '};
char mph_10[7] = {'1','0',' ','M','P','H',' '};
char mph_11[7] = {'1','1',' ','M','P','H',' '};
char mph_12[7] = {'1','2',' ','M','P','H',' '};
char mph_13[7] = {'1','3',' ','M','P','H',' '};
char mph_14[7] = {'1','4',' ','M','P','H',' '};
char mph_15[7] = {'1','5',' ','M','P','H',' '};
char mph_16[7] = {'1','6',' ','M','P','H',' '};
char mph_17[7] = {'1','7',' ','M','P','H',' '};
char mph_18[7] = {'1','8',' ','M','P','H',' '};
char mph_19[7] = {'1','9',' ','M','P','H',' '};
char mph_20[7] = {'2','0',' ','M','P','H',' '};
char mph_21[7] = {'2','1',' ','M','P','H',' '};
char mph_22[7] = {'2','2',' ','M','P','H',' '};
char mph_23[7] = {'2','3',' ','M','P','H',' '};
char mph_24[7] = {'2','4',' ','M','P','H',' '};
char mph_25[7] = {'2','5',' ','M','P','H',' '};
char mph_26[7] = {'2','6',' ','M','P','H',' '};
char mph_27[7] = {'2','7',' ','M','P','H',' '};
char mph_28[7] = {'2','8',' ','M','P','H',' '};
char mph_29[7] = {'2','9',' ','M','P','H',' '};
char mph_30[7] = {'3','0',' ','M','P','H',' '};
////////////////////////////////////////////////////////////////

char change_speed[20] = {'U','P','-','I','n','c',' ','D','N','-','D','e','c',' ','C','-','E','x','i','t'};
char change_speed_b[20] = {'B','-','C','h','a','n','g','e',' ','S','p','d',' ',' ','C','-','E','x','i','t'};
char exit[20] = {'C','-','E','x','i','t',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '};
char start[20] = {'P','r','e','s','s',' ','S','t','a','r','t',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '};
////////////////////////////////////////////////////////////////////
char input_speed[18] = {'I','n','p','u','t',' ','S','p','e','e','d',' ',' ',' ',' ','M','P','H'};
char wind_speed_2[11] = {'W','i','n','d','S','p','e','e','d',' ',' '};

int speed_value_1 = 0;
int speed_value_2 = 0;
int opt_b = 0;
int set_value;
int actual_value;
int math_value;
```

```
//////////////////////LCD Key Pad Interface Setup ////////////////////////////////////

        void USART_Init( unsigned int ubrr)

                {
                UBRR0H = 0;          // Setting Baud Rate High Registers pg 227
                UBRR0L =(unsigned char)ubrr;   // Setting Baud Rate Low Registers pg 227

                UCSR0B =(1<<RXCIE0)|(1<<RXEN0)|(1<<TXEN0); //Bit 4 Receiver Enable and Bit
                3 Transmitter Enable

                // Set Frame format: 8 data, 2 stop bits
                UCSR0C =(3<<UCSZ00);
                }

////////////////////////// Transmiter Setup ////////////////////////////////

        void USART_Transmit(unsigned char data)
        {
                        //Wait for empty transmit buffer

                while(!(UCSR0A & (1<<UDRE0)))
                {
                        ; //Do nothing
                }
                UDR0 = data; //Put data into buffer, sends the data

        }

////////////////////////////// Receiver Setup ////////////////////////////////

        unsigned char USART_Receive(void)
        {
                while( !(UCSR0A & (1<< RXC0)))
                {       ;

                }
                return UDR0; // Return the value inputted

        }

//////////////////////////Timer Setup //////////////////////////////

        void init_clk_timers()
                { //Timer set for max wind speed of 17mph
                TCNT1 = 0;
                TCCR1A=0x00;
```

```
        //Mode Slection and Prescaler settings of 8Mhz/2(65535) = 61.0361 =
16.3838ms
        TCCR1B=(0<<CS12)|(0<<CS11)|(1<<CS10);

        // TOV TIMER/CNTER OVERFLOW FLAG.
        //ICF TIMER/COUNTER INPUT CAPTURE FLAG.
        TIFR1=(1<<TOV1)|(1<<ICF1);

        // TOIE--TIMER/COUNTER OVERFLOW INTERRUPT ENABLE.
        // ICIE - TIMER/ COUNTER INPUT CAPTURE INTERRUPT ENABLE
        TIMSK1= (1 <<TOIE1)|(1 <<ICIE1);

        DDRD &=~(1<<PD4);
        }

//////////////////////MAIN//////////////////////

                int main(void)
        {       init_clk_timers(); //Initiate times
                sei(); // initiate Interrupts
                USART_Init(MYUBRR); // goes back and set the frame up with my brad
        value

                //Clear Screen
                USART_Transmit(0xFE);
                USART_Transmit(0x58);

                DDRA = 0xFF; //Data direction register port A set to output LEDs!!!
                DDRC = 0xFF; //Data direction register port C set to output to the AFD
                PORTC = ~0x40; // make sure that the AFD is off
                int switch1_flag = 0; //- Used to enable/disable
                int switch2_flag = 0; //- Used to enable/disable
                int count = 0; //- Used to enable/disable

        ////////////////////////////////////////////////////////
                //PRINT-Cal Poly
                for(i=0;i<20;i++)
                {       USART_Transmit(calpoly[i]);
                }
                        //Next Line
                        USART_Transmit(0xFE);
                        USART_Transmit(0x47);
                        USART_Transmit(1); // Colum
                        USART_Transmit(3);//Row
```

```
                    // PRINT-Learn by doing
                    for(i=0;i<20;i++)
                    {       USART_Transmit(lbd[i]);
                    }

                    //Delay for 3 seconds
                    _delay_ms(3000);

        while(1) // Forever While loop
        {
                    //Clear Screen
                    USART_Transmit(0xFE);
                    USART_Transmit(0x58);


////////////////////////////////////////////////////PrintOptions//////////////////////////////////
                                // PRINT Select Option
                                        for(i=0;i<20;i++)
                                        {       USART_Transmit(Sel_Opt[i]);
                                        }
                                //PRINT Input Wind Speed
                                        for(i=0;i<20;i++)
                                        {       USART_Transmit(Options_B[i]);
                                        }
                                //PRINT Measure Wind Speed
                                        for(i=0;i<20;i++)
                                        {       USART_Transmit(Options_A[i]);
                                        }
                            //PRINT Measure Turbine V/I/P for the future measurements
                                        for(i=0;i<20;i++)
                                        {       USART_Transmit(Options_C[i]);
                                        }



    ////////////////////////////////////EndOptions/////////////////////////////////////////////

                                menu_flag =1; // Goes into the while
                                while (menu_flag==1) // Waits for an option A,B,C
                                {       //////Button A////////
                                        if(press_function =='P')
                                        {       opt_A =0;
```

```
                    while(opt_A == 0)
                    {       // Clear Screen
                            USART_Transmit(0xFE);
                            USART_Transmit(0x58);
                            PORTA = ~0xF0; // LEDS

                            //PRINT Press Start to Turn on Fan
                            for(i=0;i<20;i++)
                            {   USART_Transmit(start[i]);
                            }_delay_ms(500);

if(press_function =='S') //Turns on the AFD
                    {       PORTA = ~0x80;
                PORTC= ~0x80; // 24V to S1 to turn on VFD
                            switch1_flag = 1;
                            // Clear Display
                            USART_Transmit(0xFE);
                            USART_Transmit(0x58);

                            while(switch1_flag ==1)
                { total_2 =avg_pw*2; // Records the Wind Speed
                PORTA = ~total_2; // Outputs on LED's

        /////////EXIT, UP, DWN, STOP COMMANDS///////

                            //Exit out of Option A button C
                                    if(press_function =='R')
                                    {       opt_A = 1;
                                            menu_flag = 0;
                                            PORTA= ~0x00;
                                            PORTC = ~0x40;
                                            switch1_flag =0;
                                    }


                            //Turns off the AFD/the button B
                            else if(press_function =='Q')
                            {       PORTA = ~0x00;
                                    PORTC = ~0x40;
                                    switch1_flag =0;
                                    _delay_ms(2000);
                            }

                            //Increase Frequency UP----
```

```
else if(press_function =='M')
{       DDRH = 0xFF;
        count=0;
while(count != 1) // increment by 1
        {       PORTA= ~0x82;
                PORTC = ~0x82;
                PORTH = ~0x00;
                _delay_ms(100);
                PORTA= ~0x80;
                PORTC = ~0x80;
                PORTH = ~0x02;
                _delay_ms(100);
                PORTH = ~0x00;
                _delay_ms(100);
                count++;
                press_function =' ';
                switch1_flag =1;
        }

        DDRH = 0x00;
}

//Decrease Frequency DWN-----

else if(press_function =='N')
{       count=0;
        DDRH = 0xFF;
while(count != 1) ) // increment by 1
        {       PORTA= ~0x81;
                PORTC = ~0x81;
                PORTH = ~0x00;
                _delay_ms(100);
                PORTA= ~0x80;
                PORTC = ~0x80;
                PORTH = ~0x01;
                _delay_ms(100);
                PORTH = ~0x00;
                _delay_ms(100);
                count++;
                press_function =' ';
                switch1_flag =1;
                }

                DDRH = 0x00;
```

```
    ////////////////////////////////WIND SPEED LOOK UP////////////////////////////////
//Clear Screen

    USART_Transmit(0xFE);
    USART_Transmit(0x58);

//Prints Wind Speed
    for(i=0;i<10;i++)
    {       USART_Transmit(wind_speed[i]);
    }
    // Row 1 Column 12
    USART_Transmit(0xFE);
    USART_Transmit(0x47);
    USART_Transmit(12); //column
    USART_Transmit(1);//row

                                    //5MPH
                                        if( (total_2<0x4A7680)  && (total_2>0x3548A0))
                                        {       for(i=0;i<7;i++)
                                                {       USART_Transmit(mph_5[i]);
                                                }
                                                        //Fourth Line
                                                        USART_Transmit(0xFE);
                                                        USART_Transmit(0x47);
                                                        USART_Transmit(1);
                                                        USART_Transmit(4);
                                                        for(i=0;i<20;i++) //Print Increase &
    Decrease Speed and Exit
                                                        { USART_Transmit(change_speed[i]);
                                                        }
                                                        _delay_ms(500);
                                        }



//6MPH

else if( (total_2<0x3548A0)  && (total_2>0x295220))
{   for(i=0;i<7;i++)
    {       USART_Transmit(mph_6[i]);

    }
    //Fourth Line
    USART_Transmit(0xFE);
```

```
        USART_Transmit(0x47);
        USART_Transmit(1);
        USART_Transmit(4);
         for(i=0;i<20;i++)
         { USART_Transmit(change_speed[i]);
         }
         _delay_ms(500);
}

.
.
.
.
.
.

//29MPH
else if( (total_2<0x6F730)  && (total_2>0x6B5F8))
{       for(i=0;i<7;i++)
        {       USART_Transmit(mph_29[i]);
        }
        //Fourth Line
        USART_Transmit(0xFE);
        USART_Transmit(0x47);
        USART_Transmit(1);
        USART_Transmit(4);
        for(i=0;i<20;i++)
        { USART_Transmit(change_speed[i]);}
        _delay_ms(500);
}


    //30MPH
    else if( (total_2<0xA8C00)  && (total_2>0x9FB78))
    {   for(i=0;i<7;i++)
        {       USART_Transmit(mph_30[i]);
        }
        //Fourth Line
        USART_Transmit(0xFE);
        USART_Transmit(0x47);
        USART_Transmit(1);
        USART_Transmit(4);
        for(i=0;i<20;i++)
```

```
{ USART_Transmit(change_speed[i]);
}
_delay_ms(500);
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
        }// end of while!!!!
}

    }

}
```

```
////////////////////////////////////OPTION B INPUTSPEED//////////////////////////////////////
        else if(press_function =='Q')
        {       // Clear Screen
                USART_Transmit(0xFE);
                USART_Transmit(0x58);
                //print input wind speed
                for(i=0;i<18;i++)
                { USART_Transmit(input_speed[i]);
                }
                opt_b =0;
                //Fourth Line
                USART_Transmit(0xFE);
                USART_Transmit(0x47);
                USART_Transmit(14);
                USART_Transmit(1);
                USART_Transmit(0xFE);
                USART_Transmit(0x4A);


        //Records 1st value
        speed_value_1 = 1;
        while( speed_value_1 == 1 )
                {       if(press_function == 'L')
                        {       speed_value_1 =0;
                                set_value =0;
                                USART_Transmit('0');
                        }

                        else if(press_function == 'A')
                        {       speed_value_1 =0;
                                set_value =10;
                                USART_Transmit('1');
```

```
        }

        else if(press_function == 'B')
        {       speed_value_1 =0;
                set_value =20;
                USART_Transmit('2');

        }

        else if(press_function == 'C')
        {       speed_value_1 =0;
                set_value =30;
                USART_Transmit('3');
        }

        else if (press_function == 'D')
        {       speed_value_1 =0;
                set_value =40;
                USART_Transmit('4');

        }

}
_delay_ms(500);
press_function = 'Z'; // Dummy Variable
speed_value_2 =1;

                        while( speed_value_2 == 1 )
                                                { if(press_function ==
                        'A')
                                                        {
                                speed_value_2 =0;

                        set_value =set_value + 1;
                        USART_Transmit('1');}

                        else if(press_function == 'B')
                        {       speed_value_2 =0;
                                set_value =set_value + 2;
                                USART_Transmit('2');

                        }

                        else if(press_function == 'C')
                        {       speed_value_2 =0;
                                set_value =set_value + 3;
                                USART_Transmit('3');
```

```
                                        }

                                        else if (press_function == 'D')
                                        {       speed_value_2 =0;
                                                set_value =set_value + 4;
                                                USART_Transmit('4');}

                                        else if(press_function == 'F')
                                        {       speed_value_2 =0;
                                                set_value =set_value + 5;
                                                USART_Transmit('5');

                                        }
                                        else if(press_function == 'G')
                                        {       speed_value_2 =0;
                                                set_value =set_value + 6;
                                                USART_Transmit('6');
                                        }

                                        else if (press_function == 'H')
                                        {       speed_value_2 =0:
                                                set_value =set_value + 7;
                                                USART_Transmit('7');
                                        }

                                        else if(press_function == 'I')
                                        {       speed_value_2 =0;
                                                set_value =set_value + 8;
                                                USART_Transmit('8');
                                        }

                                else if(press_function == 'K')
                                {       speed_value_2 =0;
                                        set_value =set_value + 9;
                                        USART_Transmit('9');

                                }
                                else if(press_function == 'L')
                                {       speed_value_2 =0;
                                        set_value =set_value + 0;
                                        USART_Transmit('0');
                                }
        }

USART_Transmit(0xFE);
USART_Transmit(0x4B);
```

```
USART_Transmit(0xFE);
USART_Transmit(0x47);
USART_Transmit(1); //column
USART_Transmit(2);//row

for(i=0;i<20;i++)
{ USART_Transmit(start[i]);
}
//////////////////////LOOK UP FOR SPEED OF WIND//////////////////////////////

while(opt_b == 0)
{       if(press_function =='S')//Turns on the AFD
        {       PORTA = ~0x80;
                PORTC= ~0x80; // 24V to S1 to turn on AFD
                switch1_flag = 1;
                while(switch1_flag ==1)
                {       total_2 =avg_pw*2; // Records the Wind Speed
                        PORTA = ~total_2; // Output to LEDS
                        //Exit out of Option A button C
                         if(press_function =='Q')
                        {       opt_b = 1;
                                switch1_flag =0;
                        }

                           ////////////////////Increments of 1////////////////////
                        // Row 1 Column 12
                        USART_Transmit(0xFE);
                        USART_Transmit(0x47);
                        USART_Transmit(1); //column
                        USART_Transmit(2);//row

                        //Prints Wind Speed
                        for(i=0;i<11;i++)
                        { USART_Transmit(wind_speed_2[i]);
                        }

                        // Row 1 Column 12
                        USART_Transmit(0xFE);
                        USART_Transmit(0x47);
                        USART_Transmit(12); //column
                        USART_Transmit(2); //row


                //5MPH
                if( (total_2<0x4A7680)  && (total_2>0x3548A0))
```

```
{       for(i=0;i<7;i++)
        {USART_Transmit(mph_5[i]);
        }

        actual_value = 5;
        //Fourth Line
        USART_Transmit(0xFE);
        USART_Transmit(0x47);
        USART_Transmit(1);
        USART_Transmit(4);

        for(i=0;i<20;i++)
        { USART_Transmit(change_speed_b[i]);
        }
        _delay_ms(500);

}
.
.
.
.
.
.
        //20MPH
        else if( (total_2<0xA8C00)  && (total_2>0x9FB78))
        {       actual_value = 20;
                for(i=0;i<7;i++)
                {       USART_Transmit(mph_20[i]);
                }
                //Fourth Line
                USART_Transmit(0xFE);
                USART_Transmit(0x47);
                USART_Transmit(1);
                USART_Transmit(4);

                for(i=0;i<20;i++)
                { USART_Transmit(change_speed_b[i]);
                }
                _delay_ms(500);

        }
```

////////////////////////////// Incremental Part//////////////////////////////////

```
    _delay_ms(1500);

    // Math part
            math_value = set_value/actual_value;
            if(press_function =='R')
            {       opt_b = 1;
                    menu_flag = 0;
                    PORTA= ~0x00;
                    PORTC = ~0x40;
                    switch1_flag =0;
                    press_function =' ';

            }

            else if( actual_value == set_value) //Desire Value is reached
            { _delay_ms(1500); // do nothing
            }


// decrease freqency
else if( actual_value > set_value)
{   count=0;
    DDRH = 0xFF;
    while(count != 1)
            {       PORTA= ~0x81;
                    PORTC = ~0x81;
                    PORTH = ~0x00;
                    _delay_ms(100);
                    PORTA= ~0x80;
                    PORTC = ~0x80;
                    PORTH = ~0x01;
                    _delay_ms(100);
                    PORTH = ~0x00;
                    _delay_ms(100);
                    count++;
                    press_function =' ';
                    switch1_flag =1;
            }
DDRH = 0x00;
    }

            //increase frequency
            else if( math_value > 2)
            {       DDRH = 0xFF;
```

```
                    count=0;
                    while(count != 5)
                    {PORTA= ~0x82;
                    PORTC = ~0x82;
                    PORTH = ~0x00;
                                                _delay_ms(100);
                    PORTA= ~0x80;
                    PORTC = ~0x80;
                    PORTH = ~0x02;
                    _delay_ms(100);
                    PORTH = ~0x00;
                    _delay_ms(400);
                    count++;
                    press_function =' ';
                    switch1_flag =1;

            } DDRH = 0x00;
}


    else if( math_value == 2)
    {       DDRH = 0xFF;
            count=0;
            while(count != 3) // Increment by 3
            {       PORTA= ~0x82;
                    PORTC = ~0x82;
                    PORTH = ~0x00;
                    _delay_ms(100);
                    PORTA= ~0x80;
                    PORTC = ~0x80;
                    PORTH = ~0x02;
                    _delay_ms(100);
                    PORTH = ~0x00;
                    _delay_ms(100);
                    count++;
                    press_function =' ';
                    switch1_flag =1;
            }
            DDRH = 0x00;
    }
    else if(math_value == 1)
    {       DDRH = 0xFF;
            count=0;
```

```
                while(count != 1
                {       PORTA= ~0x82;
                        PORTC = ~0x82;
                        PORTH = ~0x00;_delay_ms(100)
                        PORTA= ~0x80;
                        PORTC = ~0x80;
                        PORTH = ~0x02;
                        _delay_ms(100);
                        PORTH = ~0x00;
                        _delay_ms(100);
                        count++;
                        press_function =' ';
                        switch1_flag =1;

                }
                DDRH = 0x00;

    }

else if( (math_value < 2) && (math_value > 1))
{       DDRH = 0xFF;
        count=0;
        while(count != 1)
        {       PORTA= ~0x82;
                PORTC = ~0x82;
                PORTH = ~0x00;
                _delay_ms(100);
                PORTA= ~0x80;
                PORTC = ~0x80;
                PORTH = ~0x02;
                _delay_ms(100);
                PORTH = ~0x00;
                _delay_ms(100);
                count++;
                press_function =' ';
                switch1_flag =1;

        }
                DDRH = 0x00;

}
                                        }
                                }

                        }
```

```
                }

        }

}

return 0;


} // END MAIN

    // Interrupts for to receive a key value from the key pad

    ISR(USART0_RX_vect)
    {   press_function = UDR0;
    }


    // The MATH and Interrupt for each rising edge the anemometer produces

    ISR(TIMER1_CAPT_vect)
    {   total_1 = gvintT1Count * Frequency_top;
        pw = ICR1 + total_1;
        data[record] = pw;
        record++;
        TCNT1 = 0;
        gvintT1Count= 0;
        total_1=0;
        record = record%20;

        //Taking the Average

        pw_total=
        data[0]+data[1]+data[2]+data[3]+data[4]+data[5]+data[6]+data[7]+data[8]+data[9]+
        data[10]+data[11]+data[12]+data[13]+data[14]+data[15]+data[16]+data[17]+data[18]+
        data[19];


        avg_pw = pw_total / 20; // Final Value the code uses

    }

    // Interrupt when timer overflows it counts , used for the math part of the code in the
    TIMER1_CAPT_Vect
    ISR(TIMER1_OVF_vect)
    {   gvintT1Count++;
    ////////////////////////////////////////END OF CODE////////////////////////////////////////
```

Appendix C Analysis of Senior Project Design:

Project Title: Development of Wind Tunnel for Laboratory Wind Turbine Testing

Danny Zepeda:

Student's Signature: _____

Advisor's Name: Dale Dolan

Advisor's Initials:_____ Date:_____

• **Summary of Functional Requirements**
Use a micro controller to interface with LCD display, 14-key Key Pad to select wind speed and display wind speed. Use an anemometer to measure wind speed and use a variable frequency drive to turn on a duct fan to simulate wind, increase and decrease wind speed. These functions will be used to establish a wind simulator for a future lab experiments.

• **Primary Constraints**
Using low voltage micro controller with high voltage equipment like a 5HP motor and a Variable Frequency Drive, because it causes Electromagnetic Interference (EMI) which interfere with the electronics and affects the wind measurements. This took a lot of trouble-shooting and used methods to minimize EMI.

• **Economic :**

Originally this project was estimated to be less than $6,000:

Item	**Estimated Cost**
Tube Axial Fan	$3000
Induction Motor	$500
Tube Framing (Steel)	$1000
Round to Square Duct	$700
Variable Frequency Drive	$200
Other Minor Components	$500
Total:	$5900

Item	Cost
(2) Tube Axial Fan 42inch	$2,122.20
(2) Sheave, Two Grove	$49.61
(2) Bushing, Split Taper	$12.35
(3) V Belt 77 inch	$14.20
(4) Fan Guard	$ 523.36
(2) Variable Frequency Drive	$200.00
(5) 2 x 2 x .250 H.S.T. 20'	$357.60
(3) ¼ X 3" H.R. Flat 96"	$25.00
(4) 2x2X 1/8" H.R. Angle 20'	$47.19
(1) 5hp Induction Motor	**Donated**
(5) Round to Square Duct	**Donated**
Total :	3,351.51

- **Environmental:**

This Project has no environmental impact

- **Manufacturability:**

Manufacturing the steel frame was difficult since it must support a large, heavy fan and also be mobile. A good amount of planning and measurements around the EE department were made in order to create a frame that would fit in a number of rooms in the EE department.

- **Sustainability**

The purpose of the project is to provide and simulate wind and have a wind turbine produce power. The motor at full speed takes 3,730W to power and the desired wind turbine at max power would range from 200W – 450W. So it's obvious that this project will not be meant for generating power because it's not at all efficient. It is instead designed to be used for testing purposes. During the life of the project many parts were salvaged from around the EE department and free parts helped keep the project below the set budget.

Upgrades can be made by making a tight seal box to allow flow of the wind. Installing a honeycomb grid will also straighten out the air flow to accurately run wind testing.

- **Ethical**

The purpose of this project is to help develop an understanding of other methods of alternative energy and prepare future engineering students to help find alternative methods of energy generation. This project will also save the EE department the expense of purchasing a wind tunnel.

- **Health and Safety**

This project requires high voltages to power a 3phase 240 V Induction Motor. Possible cause of electrical shock if not properly used. DO NOT OPERATE ALONE.

- **Social and Political:**

No social or political problems involved.

- **Development**

Usage of a STK600 microcontroller, Software program Google Sketch, Usage of a Variable Frequency Drive , Using a RS232 cable and USART. Interfacing with a keypad. Knowledge of stick welding and steel tubing. Use of an Anemometer.